NASA-CR-193050

NAS

7N-61-CR
163066

Research On Advanced Engineering Software For In-Space Assembly and the Manned Mars Spacecraft

Final Report to the Office of Exploration

June 30, 1989

Prepared by:

Rockwell International Corporation
Space Transportation Systems Division
Expert Systems Application Group
Downey, California

Prepared for:

National Aeronautics and Space Administration
Ames Research Center
Moffett Field, California

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

Preface

This report is the culmination of research and investigation into the applicability of intelligent software for advanced exploration-class manned space missions. It is written with both the novice and advanced artificial intelligence reader in mind. It should be of interest both to researchers in artificial intelligence and to aerospace engineers investigating advanced space missions.

I owe a great debt to many people who assisted me with this work. First, I would like to thank Peter Friedland of NASA Ames, who had the foresight to see the importance of this research and initiated the funding for this report. Thanks also to Michael Sims of NASA Ames, who provided knowledgeable comments early in the project that helped to focus the report. Secondly, I owe many thanks to both Ken Fertig and Corinne Ruokangas of the Rockwell Science Center Palo Alto Laboratory, who provided insightful comments on earlier drafts of this report. I would also like to thank Nils Nilsson and Ewald Heer for taking time out of their busy schedules to answer questions and provide feedback on ideas.

Finally, I would like to acknowledge the research team who supported the generation of this report: John Hoey, John Irwin and Tom Reid. This team spent many long hours in brainstorming sessions and concept fine-tuning. They also participated in a seemingly endless number of section review meetings with patience and helpful comments.

Don DeLaquil
Project Lead
Expert Systems Application Group

Executive Summary

This report describes current and potential advanced software architectures to support space operations within the in-space assembly and manned Mars spacecraft domains. In particular, the purpose of this report is to define applications of a specific form of advanced software called intelligent communicating agents. Our approach was to first define the technological barriers to objectives within these domains and then to present potential advanced software solutions to those barriers including evaluation of intelligent communicating agents with respect to the specific domains.

Our study has shown that advanced software applications for the in-space assembly domain are limited in the time frame of launching the first manned Mars spacecraft. This is partly due to the availability of Space Station personnel for performing assembly operations and due to the prohibitive cost of developing advanced robotic systems. In addition, the current level of sensor software and hardware technology was found to be too immature to support robotic applications for intelligent communicating agents.

Within the manned Mars spacecraft domain, it was found that spacecraft operations did not *require* the full sophistication that an intelligent communicating agent software architecture might provide. Instead, it was found that there is a strong need for advanced software to support spacecraft autonomy and subsystem health maintenance. A detailed advanced software architecture is presented that could automate spacecraft operations, perform subsystem health maintenance and fault detection, isolation and reconfiguration, and provide some level of decision support for its crew when communication with mission control is not possible or is time prohibitive.

We conclude then, that within the time frame of the first manned Mars mission, full intelligent communicating agent software technology is not required for these two domains. The major reasons are:

1) There is not sufficient general-purpose requirements in either domain to warrant it.
2) There are significant software barriers that must be overcome before it is possible to field operational intelligent communicating agent systems.

The basic software barriers that must be overcome are integrating sensor software and hardware technology and implementing real-time reactive software that can also exhibit general purpose problem solving behavior.

Another conclusion that can be drawn from this study is that as exploration-class missions become more advanced and autonomy requirements increase, intelligent communicating agent software technology could become an important factor in determining the types of missions that may be attempted. Indeed, as manned and unmanned space exploration becomes more remote and autonomous, there will be a greater need for the reactive general problem solving capability that would be exhibited by operational intelligent communicating agent systems. For this technology to be realizable when other hardware and software technology matures, research into solving the barriers limiting operational intelligent communicating agent systems should begin now. A further benefit of this research would be the necessary integration of disparate artificial intelligence disciplines, resulting in a synergism and advancing artificial intelligence software technology to a level that should enable it to solve some of the challenges of space exploration missions in the 21st century.

1. Purpose and Scope

The purpose of this report is to identify potential Advanced Engineering Software (AES) applications for future NASA space missions and objectives. In particular, this report will focus on those AES applications relevant to the two mission areas of in-space assembly and manned Mars spacecraft (Code Z Scenario 2). Identification of applications that can use AES will be based on the capacity of AES to provide cost effective or improved mission-success solutions to expected technological or operational barriers. One kind of AES that will be investigated in some detail is a form of intelligent distributed software processing called Intelligent Communicating Agents (ICA).

Stanford University, SRI International and Rockwell International Science Center are currently investigating architectures for the software implementation of ICAs. The general theme of their research is to define methods for *cooperative* problem solving. Cooperative problem solving is viewed as a high-level interaction among software agents. This interaction is necessary since, in real world environments, ICAs will need to pool their knowledge by inquiring about the state of the world from other agents, and motivating other agents with different capabilities to contribute to their own problem solving.

This report augments the above research by investigating the practical applications of this ICA technology in relation to NASA's mission objectives. Though ICA technology is still in its infancy, it is possible to postulate the most effective use of this technology once it is mature. Recommended utilization of ICA technology will be constrained to those areas where ICAs are deemed necessary to meet a mission objective or to overcome a barrier to mission success. If either criterion can be met, then an analysis and justification for their usage will be presented. The central theme of this presentation will be to show why it is advantageous to use ICA technology as opposed to another AES approach. If ICA technology does not meet either criterion, then presentation of an alternative AES architecture that does meet these criteria will be provided.

Section 2 defines terminology used throughout the report. Section 3 describes mission objectives and operational scenarios for both in-space assembly and the manned Mars mission. Section 4 discusses specific barriers to the mission objectives outlined in Section 3 and provides AES solutions for these barriers. Section 5 expands on the AES solution for the manned Mars spacecraft introduced in Section 4 and presents a detailed preliminary software architecture. Section 6 focuses on ICA implementation issues and briefly describes software technology advances required for their successful implementation. Section 7 summarizes conclusions reached in this report. Section 8 contains three appendices, most notably Appendix 8.2, which describes preliminary software requirements for eight NASA identified space operations areas that warrant further study.

2. Background

Section 2.1 introduces terminology and concepts used throughout this report. Definitions for the terms artificial intelligence, expert systems, intelligent agents, distributed artificial intelligence and intelligent communicating agents are given. These definitions are provided to assist the reader in understanding the different AI approaches to complex problem solving and to show the progression from less complex to more powerful problem solving paradigms.

This report focuses on AES technology which is dependent for its operation on both AI software technology, such as sensor data fusion and interpretation, and on hardware technology, such as sensors and effectors. To provide a perspective from which to analyze potential AES applications, Section 2.2 presents a definition of two technological time frames for both in-space assembly and the manned Mars spacecraft based on current and projected software/hardware capabilities. This report will focus only on the first of the two time frames which describes technology available or achievable within current capabilities.

2.1 Terminology

2.1.1 Artificial Intelligence (AI)

As a software engineering discipline, AI consists of programs that process symbolic data as well as numeric data. When AI software is utilized to implement advanced systems for space application it is termed "Advanced Engineering Software" (AES) by the NASA Ames Research Center. Since the scope of this report is within space applications, the term AES will be used synonymously with AI. Within the field of AI/AES, some of the subfields are:

1) AI Problem Solving Techniques
   a) Exploring Alternatives (Searching)
   b) Control Strategies
   c) Expert Systems
   e) Plan Generation
   f) Reasoning Approaches
      i) Non-monotonic
      ii) Probabilistic
      iii) Theorem Proving (Predicate Calculus)
      iv) Common Sense
      v) First Principles
2) Language Understanding/Speech Recognition
3) Robotics/Image Understanding
4) Learning
5) Neural Nets

These AI subfields may each be considered separately as their own discipline or in combination when developing an integrated system. For example, many expert systems have some form of a natural language interface by which the user communicates with the expert system. In robotics applications there is often a need for a vision system to guide the use of effectors and perhaps a speech recognition system to accept commands. Generally, AES results from the integration of these subfields to generate a robust intelligent system capable of operating in complex real world environments.

2.1.2 Expert Systems

Expert Systems is a branch of AES which involves capturing a human expert's knowledge into a software program which then performs similarly to the expert in that specific domain. An expert system is comprised of a knowledge base and an inference engine. The knowledge base contains the expert's empirical knowledge (heuristics), associated procedural knowledge and a workspace containing facts which describe the current state of the problem solution. The knowledge base is frequently represented by if-then rules and facts. The inference engine compares these rules with the current problem solution state to determine which rule should be invoked next. This process is repeated in an effort to converge toward a problem solution. This process is termed "inferencing" and is generally how expert systems solve problems. Expert systems are sometimes called knowledge-based systems because their operations are based on explicit representation of the expert's knowledge.

Commercially available expert systems are usually small stand-alone systems which are too slow for critical real-time performance. These expert systems are also fragile, in that they begin to fail rapidly when used at the limits of their knowledge. NASA's requirement for intelligent autonomous systems for future space missions demands more advanced software systems which can operate autonomously in real-time [5].

2.1.3 Intelligent Agents

An intelligent agent is an artificial intelligence system that exhibits autonomous real-time behavior in complex environments. This requires perception of environmental signals and/or data, goal-directed reasoning to compute and execute appropriate actions in real-time and the ability to react to and plan around unpredictable events.

Several software architectures are currently under investigation for implementation of intelligent agents [39]. One of the most promising intelligent agent software architecture identified so far is the Belief-Desire-Intention (BDI) architecture [37]. The BDI architecture contains three main data structures:

1) Facts representing information about the world (knowledge, or more accurately, *belief*),
2) Facts representing the goals of the system (goals or *desires*),
3) Facts representing the future actions the system is considering (plans) or has decided to execute (*intentions*).

Using the BDI approach, the main components of an agent's cognitive structure are as illustrated in Figure 2.1.3. Sensed data from the external environment is processed and converted to symbolically represented beliefs about the environment. A planning system then combines information about the agent's goals (desires) and beliefs to compute candidate sets of actions. An agent commits to these actions by making appropriate changes to the "actions" module. When thus committed, the agent now has *intentions*. Actions are automatically triggered by collections of beliefs to produce signals that energize effectors. This real-time activation of effectors is also possible through use of action networks that act as "reflex arcs" to directly connect certain perceptions with effector stimulation [36].

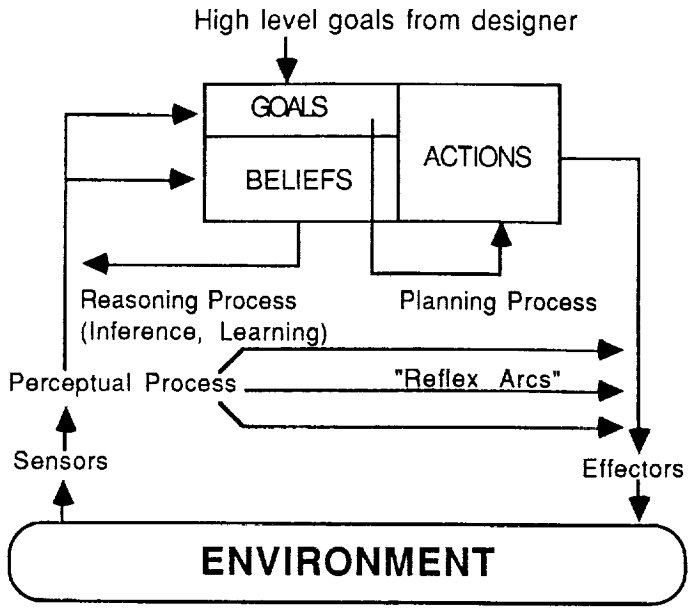


Figure 2.1.3 The Cognitive Structure of an Intelligent Agent

2.1.4 Distributed Artificial Intelligence (DAI)

A definition of a minimum DAI system is that it must include at least two agents, that these agents have some degree of information and/or control autonomy, that there is some degree of interaction between agents, and that some nonempty subset of the agents display sophistication in an AI sense (capability of reasoning, planning, etc.) [47]. A distributed approach to problem solving, under certain circumstances, can increase the capability of a computer system due to three major advantages:

1) Fail-soft degradation characteristics -- failure of one component in the system will not necessarily paralyze the entire system. Each failure may affect overall performance, but degradation of this performance will usually be gradual, and need not exhibit a sharp decline to zero.
2) Upward extensibility -- it may be possible to add processing elements to the system in an incremental fashion. This allows a gradual increase in the power of the system as additional resources become available.
3) Added capability due to the available access of specialized information (i.e., distribution of knowledge) or specialized equipment (hardware or software) -- some elements of a system may be able to run certain programs or perform certain sensory tests that cannot be done by other elements. The overall system capabilities are improved through the integration of these specialized abilities.

Certain problems lend themselves most naturally to a distributed solution. The distributed nature of particular applications can arise in a variety of distribution modès:

1) Geographic -- this is the most common basis of distribution. An example may be a distributed computer network with high data rates. The geographic distribution of the computing nodes and a high data rate argue against centralized processing, since too much data would need to be broadcast to the central node.
2) Multiple agents -- often an application requires multiple entities for a functional system. An example might be a group of robots, or airplanes, or computers.
3) Functional decomposition -- certain applications, while not strongly distributed geographically, nevertheless display a natural decomposition along functional lines. It seems most natural for certain agents to specialize in doing certain tasks, for example, as in typical office tasks or as a subsystem specialist.
4) Distributed control -- in many cases there is a desire to distribute control among various entities rather than

keeping the control centralized. An example of this might be the division of power within an organizational structure or within a spacecraft's subsystem management.

In addition to the above benefits, there are also many unsolved problems in the DAI field which current software research has only begun to address. The following list describes some of the key issues faced in DAI research: [47]

1) Global coherence out of local actions -- how to design locally motivated agents, with control over their own actions, to fulfill global system goals.
2) Organization paradigms -- how to structure a group of agents to achieve coherent interaction in terms of control and communication.
3) Distribution of control -- how should individual agents in a distributed system be controlled; central control, where one agent is in control at a time vs. autonomous control, where system control is achieved through interagent communication requests which can be refused.
4) Incomplete, possibly inconsistent, views of the world -- when agents are independently performing actions, it is unreasonable (especially in large systems) to update each agent's data base after every action is performed. Thus, it is realistic to expect variations among the world models of agents; in the extreme case agents may even have conflicting world models.
5) Communication vs. Computation -- information can be obtained in two ways: through querying of an agent who already has, or can more readily obtain, the data; or by performing the necessary computation to answer the question oneself (when possible). This is an open research topic; relevant considerations include the relative costs of the two operations, limitations on the communication bandwidth, limitations on effective processing time, concerns about communication secrecy, etc.
6) Resource trade-offs -- it is often possible to devise several plans to accomplish some goal; each of these plans may have criteria that recommend its implementation. It is usually necessary to trade off some resources in favor of others when selecting a plan for execution.
7) Synchronization of activities -- when agents are independently performing actions, there is the danger that those actions might interfere with one another. Early research has been characterized by two approaches: centrally formulating plans so that they synchronize the actions of agents, and modifying the separate, already existing plans of agents so that they do not conflict.
8) Task decomposition -- when a plan is fully developed, there remains the task of how it should be distributed through the system, i.e., what agents should carry out what parts of the plan. Ideally, the system should use information about agents and their capabilities in constructing the plan and in dividing it up into pieces for distribution.
9) Reliability and redundancy considerations -- for DAI systems to accomplish the goal of increased reliability, trade-offs must be made to determine the degree of redundancy (overlap of agent capability where possible) in agent performance. Increased reliability at the cost of increased redundancy is usually prohibitive and current research in this area is sketchy.

As this list indicates, the field of DAI is relatively new (began mid 70's) and more problems have been defined than solutions posited. Yet that too is a form of progress; there has been a subtle process of maturation in the field, even over the few years of its existence. While the above are problems in DAI, some of their solutions are addressed in current research into intelligent communicating agents.

2.1.5 Intelligent Communicating Agents (ICAs)

An ICA is an intelligent agent that is an element of a set of distributed intelligent agents which communicate with each other to collaborate in task performance [37]. ICAs are a synthesis of the flexibility of an intelligent agent with the robustness of a distributed AI system. For an ICA to cooperate effectively with other agents, each must have certain abilities that fall into the following categories:

1) Reasoning about the environment, including reasoning about the beliefs, goals (desires) and intentions of other agents.
2) Communicating to exchange information about the environment and intentions to act.
3) Reasoning about actions and events, including reasoning about the effects of actions.

4) Forming and executing cooperative plans.
5) Monitoring and synchronizing the execution of individual plans.
6) Performing reactive real-time planning of the actions it is capable of executing.

Because ICA's are a loosely coupled distributed software system, they have the advantages described above for DAI systems with the additional ability of communication with other agents. This ability allows for cooperative problem solving and the automated performance of tasks which naturally require cooperation, e.g., site identification and preparation prior to a manned mission.

2.2 Technological Time Frames

When AES for space operations become mature they will provide significant payoffs in the following areas: [5]

1) Reduced mission operations costs through automation of labor intensive operations (reduce manpower),
2) Increased mission productivity through automation of routine on-board housekeeping functions (off-load astronaut time),
3) Increased mission success probability through automation of real-time contingency replanning (save experiments or possibly entire missions).

These payoffs are realizable given sufficient advances in both software and hardware technology. Some progress has already been made in reactive planning approaches, as exemplified by the procedural reasoning system used by SRI International's Flakey the Robot [53]. The procedural reasoning system utilizes a BDI-like architecture which allows Flakey to react in real-time to a set of complex and changing environments. Current limitations to Flakey's real-time operations includes the lack of an efficient process of converting low-level video and sonar sensor data into the high-level symbolic representation required by the reactive planning system. This problem is not only specific to Flakey, but is common to many kinds of sensors that must obtain real-time feedback from their environment; the conversion of sensor data to a usable form is relatively slow in relation to constraints on real-time operations [48]. In addition, there is the "sensor fusion" problem, where sometimes multiple sensor data must be merged together to produce symbolic data interpretation. This lack of synergism between low-level and high-level data processing defines a software technology gap between these two interdependent functions. Research areas attempting to limit this gap include: sensor hardware, resource bounded problem solving methods [11], problems of implementing DAI systems [section 2.1.4], processing unexpected events in complex environments, etc.

The gap between current and future AES technology bases constrains when and what kind of operational AES systems may be fielded. In part the size of this technology gap is based upon the domain's specific software and hardware requirements and consequently is different for each of the two domains within this report. To provide a framework from which to describe potential AES applications within each domain, their hardware and software requirements/constraints will be used to define two "technological time frames". Those AES applications that could be implemented using current technology with predicted discrete advances in both software and hardware technology are denoted as "current-technology" time frame applications. Those AES applications that require a continual more unpredictable growth in software and hardware technology are denoted as "projected-technology" time frame applications. Although both time frames are of interest in defining AES applications, this report will only attempt to identify those relevant to the current-technology time frame.

2.2.1 In-Space Assembly

The major constraints for in-space assembly lies in maturity of sensor and effector technology. In particular, current software techniques for converting vision sensor data into recognizable objects are still primitive, as was demonstrated during testing of an actual autonomous land vehicle [32]. It was found that even sophisticated vision systems are easily fooled by changes in the environment caused by time, weather and shadow, and could only perform relatively simple tasks (by human standards) such as road following, landmark recognition and circumventing obstacles and hazardous terrains. It was also found that a more general vision system requires extensive world knowledge and such a system has not been developed based on current technology. Since the in-space assembly domain is heavily dependent on sensor technology, the capability of converting low-level sensor data into a high-level format usable by real-time reactive planning software demarcates the technological time frame for this domain; a limited capability of sensor/software integration denotes the current-technology time frame and a robust, general purpose capability the projected-technology time frame.

2.2.2 Manned Mars Spacecraft

Since the Mars spacecraft is manned, there is less direct interaction and dependence on sensor and effector technology than in the in-space assembly domain. The main operational constraint in this domain lies in the efficient processing of various data to support spacecraft subsystem operations. Efficient processing of data, though dependent on both hardware and software, is less dependent on the maturity of embedded hardware technology than it is on software technology. Since is difficult to gauge progress in software technology it must be assumed that AES techniques will gradually increase in power over time allowing for progressively more advanced software solutions for vehicle operations. This gradual development of software technology does not allow for a precise definition of the two technological time frames in this domain. In addition, extended duration spacecraft tend to share the same basic functional requirements regardless of their sophistication level. For example, both the proposed Mars Split/Sprint spacecraft [30] and the Mars Cycling spacecraft [35] have the same basic autonomy requirements due to their long duration missions in remote trajectories from Earth. For these two reasons, the technological time frames for this domain which should be based on software and autonomy capabilities will not be defined. To compensate for this, any AES solutions recommended for the manned Mars spacecraft will be presented with details on the software technologies required for successful implementation.

3. Mission Objectives

This section describes major mission objectives associated with in-space assembly and the manned Mars spacecraft. These objectives are first described generally and then followed with a specific scenario to highlight areas where automation and robotics are most applicable.

3.1 In-space Assembly

In-space assembly is a generic term describing the assembly of any large structure in low Earth orbit, e.g., Space Station, Mars spacecraft, space manufactured structures, etc. Since this report focuses on the current-technology time frame where the mission objective is assembly of spacecraft in orbit, certain space operations assumptions are justified. First, it is assumed that there will be a space transportation system available to place large payloads into low Earth orbit, i.e., the Space Shuttle and subsequently a heavy-lift vehicle such as the proposed Advanced Launch System or Shuttle-C. Secondly, space assembly in the current-technology time frame will be limited to connecting large pretested modules with fastener technology, performing seaming operations if necessary, and then verifying structural integrity [9]. Thirdly, extra-vehicular activities are assumed to be minimized due to inherent difficulties in handling large objects and the lack of long-term physical endurance [4].

According to a recent study [9], assembling the Mars manned and cargo spacecraft requires approximately 15 and 7 heavy-lift vehicle launches respectively. The payload from one launch should be fully assembled into the vehicle structure before the next one arrives. There are three basic assembly scenarios corresponding to the three types of structures to be assembled: aerobrakes, Crew and Cargo Ships, and Crew and Cargo Propulsion Stages. All three assembly scenarios utilize two remote manipulation arms with various tool effectors and vision systems.

In this study, assembly of both the Cargo and Crew Ships for the Mars mission would first involve assembling the aerobraking shell (used for decelerating into Mars orbit) and then utilizing it as a platform for assembling remaining parts of the spacecraft. The initial assembly of the aerobraking shell can be performed while docked to the Space Station and then released into free (co-orbiting) flight for subsequent assembly operations.

This assembly task can be performed by the coordinated effort of two 30 meter long robotic remote manipulation system arms. There will be remote manipulation arms on a Space Station truss (to assemble the aerobrakes), on the aerobrakes themselves (to assemble the Crew and Cargo Ships) and on a Construction Utility Ring (to assemble the Crew and Cargo Propulsion Stage) [9]. These robotic systems can be controlled by people via telepresence technology supported by planned and contingency extra-vehicular activities when necessary. Control of the robotic assembly task depends on a hierarchical structure of human control over robotic actions. This hierarchy would span the supervisory range from top-level commands like "Put the aerobrake sections together" all the way down to machine-level instructions on how to move one manipulator joint.

A typical assembly scenario is the assembly steps for the Mars Lander and Descent Propulsion Structure. First, a 7.6 meter diameter payload carrier containing the Mars Lander and associated systems berths directly to the aerobrake with two short holding arms. The Mars Lander is held in one remote manipulation arm while the other arm attaches its three descent propulsion pods and landing gear. This assembly is then mounted with flight-release latches onto the aerobrake.

Coordination of the remote manipulation arms can be effected through use of several positioning sensors. Video cameras allow direct intra-vehicular activity monitoring of the remote manipulation system end activities, and small embedded fisheye CCD sensors provide machine vision directly from the end effectors. All hardware used in the assembly is tagged with a barcode ID for positive machine recognition. Effectors are equipped with 6-axis electromagnetic antennas, which the robot controller uses to determine the end effector's location and orientation relative to the coordinate system established by electromagnetic beacons distributed across the site.

3.2 Manned Mars Spacecraft

NASA has defined four high-level scenarios for future manned exploration-class missions: [4]

1) Human Expedition to Phobos,
2) Human Expedition to Mars,
3) Lunar Observatories,
4) Lunar to Early Mars Outposts.

This report is relevant to the first two scenarios but is concerned most specifically with scenario two. Within scenario two it is assumed that a split/sprint mission of 440 days will be used [30]. In the split/sprint scenario, an unmanned cargo spacecraft is sent to Mars along a high velocity trajectory several months in advance of the manned spacecraft.

Our discussion will be limited to AES applications for the manned Mars spacecraft itself. In the current-technology time frame, spacecraft will be of the short duration sprint type. In the projected-technology time frame, the Mars missions will utilize cyclic spacecraft which permanently cycle back and forth between the orbits of Earth and Mars [35]. Both spacecraft will have similar autonomy requirements based on long duration missions in remote trajectories from Earth. This similar autonomy requirement precludes the need for differentiating technological time frames when describing mission objectives in this section.

The Mars sprint spacecraft consists of several functional modules: [30]

1) Mars orbiting vehicle,
2) Mars descent vehicle,
   a) Mars landing module
   b) Mars ascent vehicle
3) Trans-Mars propulsion system,
4) Trans-Earth propulsion system.

A cargo vehicle containing the trans-Earth propulsion system and Mars descent vehicle will be sent to Mars in advance of the manned vehicle containing the trans-Mars propulsion system and the Mars Orbiting Vehicle (MOV). When the MOV aerobrakes into Mars orbit it will rendezvous with the cargo vehicle already there. The Mars descent vehicle will then deorbit and remain on the surface for up to 30 days. The Mars landing module remains on the surface as the Mars ascent vehicle returns to orbit to rendezvous and dock with the MOV. After crew transfer, the Mars ascent vehicle is jettisoned and the trans-Earth propulsion system performs a departure burn to bring the MOV back to Earth. Once in Earth orbit a descent capsule will deorbit bringing the crew to the surface.

Due to the long duration utilization of the MOV it will be necessary to perform maintenance on and diagnosis of subsystem performance. Each subsystem will require some degree of software control to allow for semi-autonomous functionality that would off-load crew involvement with subsystem operations. In addition, certain subsystems will require a high degree of autonomy, i.e., Fault Detection, Isolation and Reconfiguration (FDIR), to facilitate on-board maintenance [30]. The subsystems comprising the MOV which require software control are:

1) Guidance, navigation and control,
2) Environmental control and life support system,
3) Electrical power system,
4) Thermal control system,
5) Orbital maneuvering system,
6) Reactive control system,
7) Data management system,

8) Propulsion,
9) Communications,
10) Power generation system,
11) Avionics,
12) Sensors.

At this point a detailed scenario describing a diagnostic procedure for a malfunction within a certain subsystem would be appropriate, but this is dependent on the degree of software automation resident in that subsystem. Indeed, the maintenance and diagnostic procedures that will be utilized on the MOV must themselves use the subsystem or vehicle software architecture to function. Therefore, description of a subsystem diagnostic scenario will be postponed until after the vehicle software architecture is defined in section 4.2.

4. Barriers to Mission Objectives

The technological base for space systems capable of successfully implementing a manned Mars mission is still in its infancy. Today's hardware and software technologies in automation and robotics may be obsolete by the year 2000, at the time when space systems for the exploration-class missions will be designed [4]. This gap in the current and future technology base for AES is what is meant by a barrier to mission objectives. This report must then define those technological barriers to successful implementation of manned exploration missions which can be addressed by utilizing AES. The mission objectives presented in section three will be referenced to specifically describe and justify AES solutions to technological barriers.

4.1 In-Space Assembly

Based on the current-technology in-space assembly scenario described in section three, there are several reasons why it would not be cost effective to automate this process by utilizing ICA or other AES technology hosted onto robotics:

1) The cost of robotics technology is extremely high. It is estimated that the cost for full development of one robot for space assembly tasks would be on the order of 200 million dollars [25].
2) Especially in early missions, where segments to be assembled would be lifted into low Earth orbit and connected using fastener technology, there is a lack of repetitive operations that would warrant the use of specialized robots.
3) Given a fully operational Space Station with the availability of astronauts for extra-vehicular activity assembly tasks, it is more cost effective to utilize this capability when necessary for contingency or planned operations.
4) Even with repetitive tasks, such as truss building, there is no need for generalized robotics intelligence. A specialized robot could be used that is controlled via some telepresence hardware [2].

Given the above reasons, there are no autonomous robotics requirements for this phase of the NASA mission planning. In fact, because the Mars mission will consist of large segments lifted into orbit by the Space Shuttle and/or an Advanced Launch System and then connected with fastener technology, the only foreseen difficult assembly issue involves the ablative cone (similar to what was used on the bottom of the Apollo reentry capsules) required for aerobraking into Mars orbit capture and entry.

Because this ablative cone will be 28 meters in diameter, a single solid cone is beyond the cargo capacity of either the Space Shuttle or the proposed Advanced Launch System. Therefore the cone will be lifted in pieces; once in orbit, some assembly processing will be performed. This processing can be accomplished by astronauts during extra-vehicular activities, or by utilizing specialized robotic remote manipulation arms, or a combination of both.

Though there are no immediate general purpose robotics needs in this time frame, a gradual development of in-space assembly technology from remote manipulation arms and extra-vehicular activity, to telepresence, to supervisory semi-autonomous robotic systems, to fully autonomous robotic systems is assumed. This gradual development will insure that in-space assembly techniques are mature in the projected-technology time frame to support full ICA implementations [1].

Our analysis of robotic solutions for in-space assembly indicates that there is rarely a need to automate a general solution to operational requirements given that there is a person in the loop. Without manned presence, general software solutions will be very costly, difficult to implement and test thoroughly, and should only be pursued for solving general operational requirements that persist over time. For example, the reusable resource of generic software capable of directing autonomous exploration-class planetary rovers, could warrant development of an ICA technology base. But in-space assembly operations during the current-technology time frame and perhaps well beyond will likely consist of specific requirements which are insufficient to warrant utilization of ICA technology.

4.2 Manned Mars Vehicle

Each of the many modules comprising the Mars vehicle has its own particular requirements for software control and automation. The trans-Earth propulsion system and trans-Mars propulsion system have the least AES requirements due to the current maturity of expendable propulsion technology. The Mars landing module and Mars ascent vehicle have only minor AES requirements due to the short-duration autonomous nature of their functional usage. The Mars orbiting vehicle has the highest requirements for AES since it is the focal point for mission control during Mars transit and orbit, and therefore will be the focus of this section.

During the projected 14 month manned Mars mission, MOV autonomy is necessary due to constraints limiting communications between Earth mission control and the MOV. Constraints that limit communications are numerous: planetary occultation, two-way light travel time, limited communications bandwidth, error rate, radio interference from solar flares, available power, etc. [4]. Of all these causes for minimized communications, the two-way light travel time and planetary occultation are most prominent. The communications lag of between 8.5 and 42 minutes two-way light time travel while the MOV is orbiting Mars is exacerbated by the MOV being occulted for approximately 57 minutes every time its orbit takes it behind Mars. Adding these two factors brings the total potential time delay for communications to between 66 and 99 minutes. This problem is not unique to the Mars mission; it is generic to all space missions. For example, a Lunar orbiting vehicle will be occulted by the moon every orbit for approximately 80 minutes even though the one-way light travel time to Earth is only 2.7 seconds.

AES can reduce the dangers of this communications barrier by providing several vital functions on-board the MOV, including but not limited to:

1) Vehicle subsystem fault detection, isolation and reconfiguration,
2) Real-time mission support for contingency operations.

For example, during Space Shuttle operations numerous subsystem failures have occurred which required mission support to resolve. Though these on-orbit failures to date have not been critical in nature, the number of potentially critical failures within Space Shuttle subsystems is very large. Indeed, an analysis of all potential subsystem failures for the Shuttle shows a total of 207 possible failures, and assigning a high, medium or low level of criticality to each failure, 98 of these failures (47%) fall into the high criticality level [45]. Table 4.2.1 shows that the time available for resolving these critical failures is often well under the time lost to communications delay for the MOV orbiting Mars. If the current Shuttle failure rate is indicative of typical space operations, then it is prudent to have some AES to support subsystem failure resolution. In fact, a recent study has recommended that research should begin now for development of on-board maintenance systems for long duration manned missions [30].

Table 4.2.1 Space Shuttle Potential Subsystem Failures.

Subsystem	# of High Criticality Failures	Response Time
Ammonia Boiler	1	Function of leak size
Active Thermal Control	2	Hours
Atmospheric Revitalization	5	Seconds
Smoke	1	Seconds
Electrical Power Distribution Control	63	Minutes
Fuel Cell	2	6 Minutes
Hydraulics	1 2	Minutes Hours
Reaction Control System	12	Minutes
Orbital Maneuvering System	8	Minutes
Power Reactant Storage and Distribution	1	Seconds

The need for autonomous mission support on-board the MOV is exemplified by the situation encountered by Apollo 13 when an O2 tank on the Service Module exploded [16]. Appendix 8.1 contains a detailed account of the dialogue and timelines of information that was exchanged between mission control and the disabled Apollo 13 vehicle. For purposes of illustration this dialogue is interspersed with timelines that would be valid if this were the MOV in orbit. What should also be noticed from this account is the similarity between the problem solving process used by mission control and how an AES system should operate.

NASA has stated that all future space missions will require some degree of AES support and recommends the use of expert systems for semi-autonomous and autonomous control of space systems [5]. In addition, it is understood that current rule-based expert systems are too inflexible to support the required autonomy; it will be necessary to develop diagnostic model-based systems [35] (see section 6.2.2 for a definition of diagnostic model-based systems). Indeed, a flexible decision-support and fault detection, isolation and reconfiguration AES system may require implementing a hybrid software approach integrating procedural, expert system, model-based diagnostic and reactive planning software technologies.

Resulting from research into software support of space systems such as the Space Shuttle, Space Station and the Advanced Launch System, Rockwell has developed several viable software architectures for space applications. Integrating various aspects from each of these architectures has produced a potential architecture for use in the MOV. Although this architecture is presented in detail in section 5, it should be understood as a preliminary "strawman" version of a MOV AES system.

5. AES for the Mars Orbiting Vehicle

At this time a detailed presentation of an AES system architecture for the MOV is necessarily preliminary since it is based solely on information that is currently available. As Mars exploration-class mission scenarios mature and new information becomes available, this software architecture would also change accordingly.

5.1 Software Architecture

Figure 5.1-1 depicts the high-level view of an embedded on-board AES architecture that could be applied to the MOV [40]. In this design, the User Interface program provides an interface between the user and the Vehicle Manager program. The User Interface communicates with the user through a variety of media: natural language (both written and spoken), menu options, graphics, touchpanel or any combination of these formats The format of the information displayed to the user is commensurate with the detail required for users to effectively carry out tasks given varying levels of expertise. This flexibility enables a user to interact with the system from the highest command level to the lowest component level without having to be specially trained in every aspect of the system. Although the User Interface is predominantly procedural software for controlling graphics, menus and keyboard handling, it should also contain some declarative software to process both verbal and keyboard activated natural language requests.

The central hub and most sophisticated software program in the MOV AES system architecture is the Vehicle Manager. A symbolic program with intelligent agent capabilities, it could provide user decision-support through utilizing high-level knowledge of the MOV, its mission and experiment requirements, and foreseen contingency operations. It could handle some unforeseen contingency operations through use of a database encoding examples of real world common sense knowledge combined with probabilistic reasoning and reactive planning. The Vehicle Manager interfaces with the Vehicle Control and FDIR program providing approval of any vehicle reconfiguration plans. It also interfaces with the On-Line Documentation program to preserve procedure traceability as they are utilized by the crew.

The On-Line Documentation program could provide access to all nominal and off-nominal operational procedures in the format of text scripts. These scripts would detail procedures for manual testing and repair operations, manual testing and safing operations, experiment processing, crew activity and maintenance schedules, and crew specific medical information.

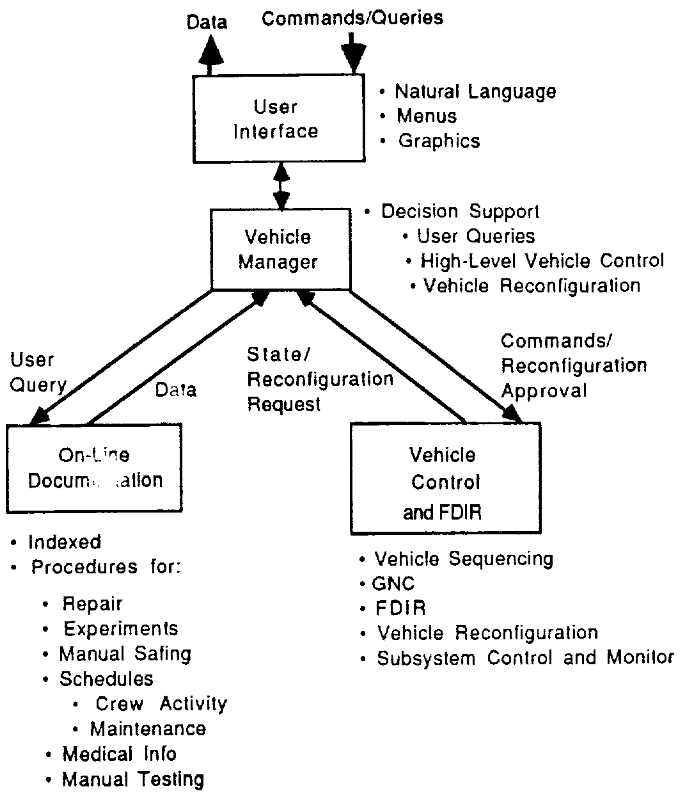


Figure 5.1-1 Block Diagram of the Embedded MOV AES System.

Figure 5.1-2 presents a more detailed view of the Vehicle Control and FDIR program architecture. It could be a hybrid software architecture combining both procedural and symbolic implementations. All nominal vehicle operations would be controlled through high-level commands issued by the Vehicle Sequencing and Guidance, Navigation and Control (GNC) programs to the Subsystem Control and Monitor programs. The Vehicle Sequencing and GNC programs are procedural software (probably Ada), that could utilize interrupt-driven and high-frequency tasked implementations, respectively. Sequencing commands should be synchronized with data from GNC, which itself would require a high-frequency procedural implementation (also probably tasked Ada). There would be a Subsystem Control and Monitor program for each subsystem that controls the subsystem hardware by executing Vehicle Sequencing or GNC commands. The Subsystem Control and Monitor program could perform low-level fault detection and trend analysis for subsystem health maintenance by monitoring built-in test equipment and sensor data. If spacecraft operations remain nominal during the Mars mission, this procedural software architecture could suffice to control subsystem operations.

When anomalous behavior is observed or predicted by the Subsystem Control and Monitor program, it could pass all relevant data to the Vehicle Fault Diagnosis Expert System (ES) for diagnostic processing. The Vehicle Fault Diagnosis ES could use these data in conjunction with knowledge of interactions and dependencies between subsystems to determine a diagnosis based on a global vehicle perspective. This fault diagnosis data can then be passed to the Vehicle Reconfiguration ES which would generate a vehicle reconfiguration command sequence to provide a fault work-around. This reconfiguration command sequence would not be immediately issued to the Vehicle Sequencing program, but first a reconfiguration request would be issued to the Vehicle Manager which decides whether or not the current reconfiguration request warrants user intervention. In either case, a reconfiguration decision would be issued back to the Vehicle Reconfiguration ES which would control transmission of reconfiguration commands to the Vehicle Sequencing program.

The Vehicle Control and FDIR program could be a hybrid KBS which utilizes both empirical heuristic knowledge extracted from subsystem experts and a model-based reasoning approach to problem solving. These two reasoning process could execute in parallel to provide an efficient diagnosis of subsystem failures. The heuristic-based reasoner could provide a fast response time to expert provided known anomalies. The model-based reasoner could diagnose anomalies which were not foreseen by subsystem experts. It would utilizes an internal model of how the subsystem operates (including its interactions with other subsystems) and could perform tests to determine what caused the difference between the expected internal state of the model and the current sensed state of the subsystem.

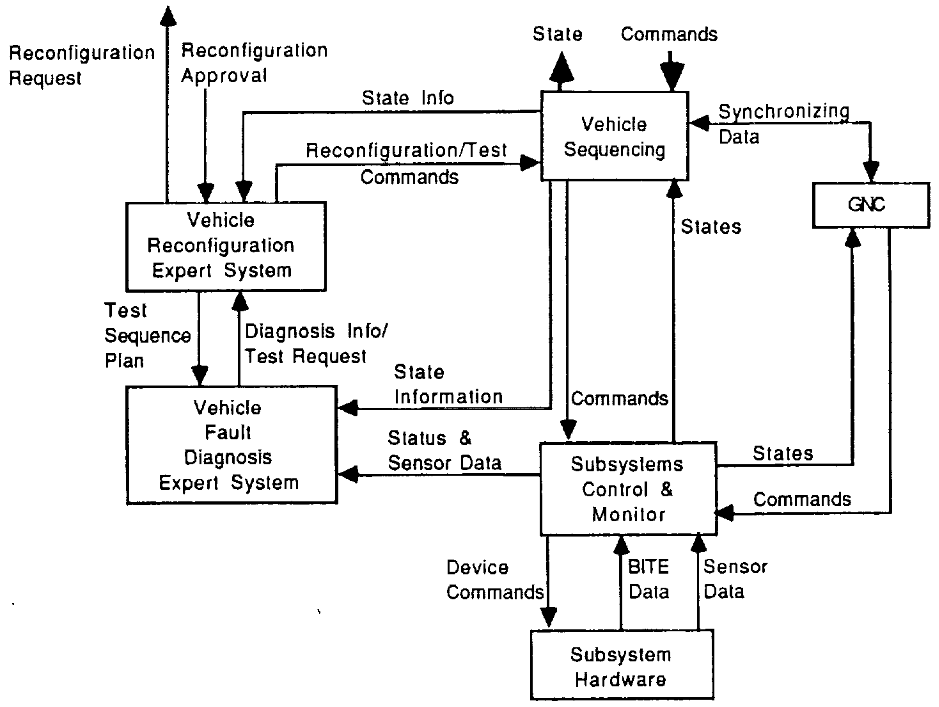


Figure 5.1-2 Block Diagram of the Vehicle Control and FDIR Architecture.

5.2 Data Management System Hardware Architecture

Due to the extended duration of the Mars mission the probability of hardware failures should be greater than what is experienced today on short duration space missions. Therefore, the design of a MOV data management system hardware architecture should consider redundancy and safety issues in addition to software execution efficiency. The data management system hardware architecture depicted in Figure 5.2 should increase system reliability through dynamic allocation of software programs and decrease cost and complexity through use of standardized hardware processors. Our description is purposely high-level to provide a generic solution to the data management system hardware architecture which could then be implemented with specific hardware.

Current studies indicate that the MOV will have at a minimum three standard Space Station habitation/lab modules [9]. The data management system hardware architecture should be duplicated to provide complete redundancy in the event that any single module becomes disabled. In addition, within each data management system there could be extra standardized processors to provide another layer of redundancy. This approach to the hardware architecture would provide two layers of safety in operations:

1) The chances that both Space Station modules hosting the data management system are disabled during the mission is remote.
2) Within each data management system hardware architecture, if a standard processor fails, its software tasks could be dynamically switched to another available processor.

An operating system could control a local area network connecting sets of standardized processors shown in Figure 5.2. The operating system could consist of: an Executive program that dynamically allocates software programs to processors based on current processor, subsystem and vehicle loads; and a local area network Controller program that would respond to low-level processor requests for use of the local area network. Multiple programs could be loaded onto an individual processor which would contain its own low-level software to control multi-tasking of programs. The local area network should be doubly redundant to provide backup in the event of a bus failure. If the active local area network should fail, network communications could be transferred to the backup local area network while the failed one, if possible, is rebooted.

The Executive program would control utilization of three types of processors: procedural, symbolic and specialized database. It would automatically load software programs onto an appropriate available processor, e.g., symbolic software to a symbolic processor, as demanded by vehicle operations. Dynamic software allocation onto multiple processors could be effective for several reasons:

1) It would eliminate the need for specialized doubly or triply redundant processors (except in the case of the data base processors) which decreases system complexity.
2) It would allow for standardized commercially available processor sets which decreases cost.
3) It would provide increased reliability since a failed processor's software could be reallocated to another available processor.
4) It would promote parallel software operations by utilizing multiple processors.

Display terminals with graphics and touchpanel capabilities should be resident in all habitation/lab modules to provide access into the data management system. Strategically placed voice recognition modules and voice synthesizers could respectively support crew voice commands to the user interface and allow the data management system to audibly annunciate critical conditions to all crew members.

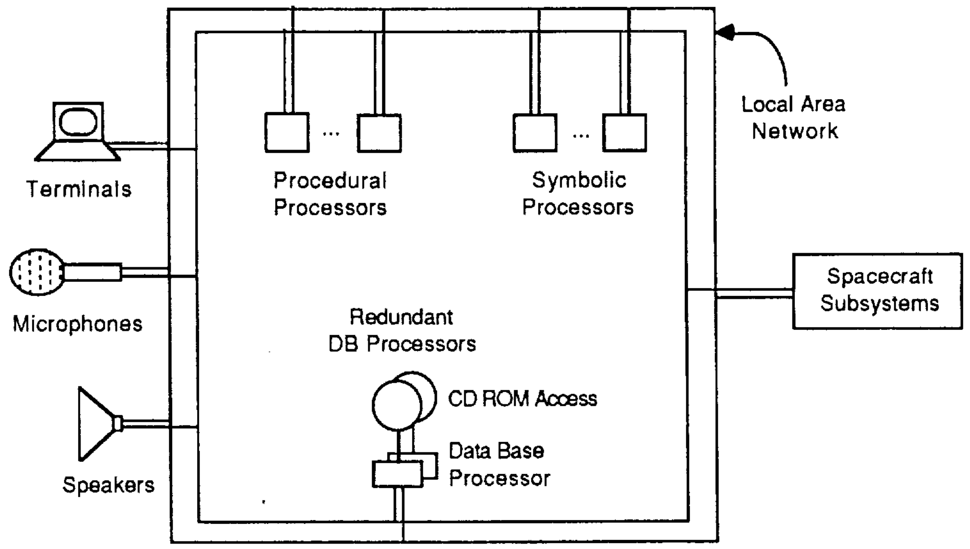


Figure 5.2 The MOV Data Management System Hardware Architecture.

Table 5.2 describes the mapping of software programs defined above to data management system hardware processors. Some programs share processors and others have a dedicated processor to facilitate parallel operations during peak vehicle load conditions. For example, the eleven subsystems comprising the MOV could each be controlled by a corresponding Subsystem Control and Monitor program that may be mapped to much fewer than eleven processors (perhaps only three) due to multiple programs executing on each processor (known as multi-tasking). Subsystem Control and Monitor program multi-tasking would be feasible due to reduced requirements for parallel subsystem operations.

Table 5.2 Data Management System Processor Types.

Software Program	Processor Type
User Interface	Procedural
Vehicle Manager	Symbolic
On-Line Documentation	Data Base
Vehicle Reconfiguration ES	Symbolic
Vehicle Fault Diagnosis ES	Symbolic
Vehicle Sequencing	Procedural
GNC	Procedural
Subsystem Control & Monitor	Procedural
Operating System	Procedural

5.3 Software Operations Scenario

The Vehicle Control and FDIR program would automatically execute all major vehicle operations, such as the trans-Mars propulsion burns and Mars aerobrake capture. Vehicle subsystem hardware would be controlled via high-level commands issued by the Vehicle Sequencing and GNC programs to the appropriate Subsystem Control and Monitor program which would then issue an equivalent set of low-level commands to the subsystem hardware. The On-Line Documentation program would provide astronauts with detailed schedules for experiments, maintenance, exercise and other crew activities. The Vehicle Manager program would notify the crew of vehicle health and the schedule for all major vehicle operations. If vehicle operations remain nominal during the Mars mission, this software set would be sufficient to successfully support vehicle operations.

In the event of a device failure on-board the MOV, other software programs would then be invoked. The Subsystem Control and Monitor programs would utilize Built-In Test Equipment (BITE) and sensor data to continuously monitor the health of all spacecraft subsystems. When an anomalous condition is observed or a failure is predicted by the appropriate Subsystem Control and Monitor program, all relevant data would be passed to the Vehicle Fault Diagnosis ES. The Vehicle Fault Diagnosis ES would utilize both heuristic and model-based reasoning to diagnose and isolate the failure to a single definite or set of probable failed devices. Fault isolation would be to the subsystem replaceable unit or to the line replaceable unit component level. These diagnostic data are passed to the Vehicle Reconfiguration ES which could generate a high-level plan based on overall vehicle functionality either to switch to a redundant unit (if it exists) or perform a subsystem reconfiguration to work-around the failed component.

Once a plan is generated, the Vehicle Reconfiguration ES would determine the failure's criticality level based on how hazardous the failure is and the available response time for failure correction. In order of decreasing criticality, there would be at least three fault levels:

1) A life-threatening situation requiring a response within seconds.
2) A hazardous condition requiring a response within minutes or greater.
3) A non-hazardous situation requiring user intervention.

In case one, the Vehicle Reconfiguration ES would immediately issue the high-level reconfiguration plan to the Vehicle Sequencing program which then would command the appropriate Subsystem Control and Monitor program to issue low-level hardware safing commands to the faulty subsystem. Once the vehicle is in a safe configuration, if the Vehicle Fault Diagnosis ES had diagnosed a set of potential faults, the Vehicle Reconfiguration ES would perform additional tests on the failed subsystem to allow the Vehicle Fault Diagnosis ES to further isolate the cause of the failure. The Vehicle Reconfiguration ES would perform these tests because only it is aware of the current vehicle configuration and is therefore able to safely time subsystem tests based on this information. Tests performed by the Vehicle Reconfiguration ES would be synchronized with the Vehicle Fault Diagnosis ES so that the Vehicle Fault Diagnosis ES could interpret test results.

In case two, the Vehicle Reconfiguration ES would issue a reconfiguration request to the Vehicle Manager. The Vehicle Manager would announce via the user interface that the component in question has failed, declare response timelines and effects of the failed component, and provide the user with options for rectifying the situation. The user would select an option and the Vehicle Manager would take appropriate actions. Some user options, in order of increasingly available response time, are:

A) Authorize the Vehicle Manager to direct the Vehicle Reconfiguration ES to issue the high-level reconfiguration plan to the Vehicle Sequencing program that would result in a work-around or switch to redundant component. For safety reasons this is the default action that would be taken if the user fails to select an option within the announced response time. This default behavior assumes that all critical vehicle failures will have a preplanned automatic contingency procedure.
B) Request the On-Line Documentation program to provide the user with instructions for manually safing the failed subsystem. Once safing is complete, further instructions would be provided for subsystem contingency operations if required.
C) Replace the failed component with a new one if available. The Vehicle Manager would direct the user to interact with the On-Line Documentation program for procedures to isolate and swap-out the component. If a new component is not available, the Vehicle Manager would direct the user to access the On-Line Documentation program for work-around procedures.

In the non-hazardous case three, the Vehicle Manager would operate essentially the same as shown in C above, with the important difference that if the user failed to take any action, the Vehicle Manager would not perform any automatic procedure since this is not considered a hazardous situation. Also in this case, the user would have the option of performing an interactive diagnosis of the fault cause by utilizing two Vehicle Manager capabilities:

1) Browsing through subsystem built-in test equipment and sensor data to determine a testing approach for analyzing the fault cause.
2) Initiate automatic or manual subsystem tests, analyze their results and perhaps request additional tests, until diagnosis of the fault cause is complete.

Regardless of which action was taken, the Vehicle Manager would notify the crew of any changes in vehicle operations due to the safing or reconfiguration of a subsystem. In addition, all vehicle reconfiguration data would be logged and periodically transmitted to Mission Control for ground-based specialist analysis.

6. ICA Implementation Issues

So far this report has talked little about software technology requisite for operational ICA software systems. This section therefore attempts to define the major software capabilities that are required for successful ICA implementations. Section 6.1 discusses general AES/ICA operational requirements and 6.2 discusses specific software technology areas deemed essential for viable ICA systems. The approach is to first provide a software capabilities baseline by defining the current state-of-the-art in AES technology and then to describe what advances from this baseline are required to support ICA implementation. The purpose of this section is not to determine what specific resources or processes are needed to overcome the differences between these two capabilities, but to simply identify them for assessment of future research objectives.

6.1 Operational Requirements

Although software architectures for ICAs are currently still in the research stage, many of their operational requirements are well understood and can be defined. An approach for defining ICA operational requirements is to first describe the computational load of current AES programs which perform tasks considered similar to ICA operations (albeit at a lesser degree of sophistication) and then define the advances from this baseline required for projected ICA performance. To provide the baseline needed for this discussion, three AES systems will be described: Hitech [8], a master's level chess program, Airtrac [34], an aircraft tracking simulation system, and SIPE [23], a robotic simulation system. These three program were selected because they exhibit capabilities that are considered essential to successful ICA systems.

Before a description of these three AES systems is presented, a short discourse on problem solving is warranted. All AES programs have in common the need to make decisions for selecting the most promising next step in solving a problem from a set of competing alternative steps. This process can be viewed as searching through a tree or network of potential steps and choosing the "best" next step. Searching is critical to real-time problem solving due to the large number of potential steps that must be chosen from during the problem solving process. For example, if each decision point in a network has five branches and the network is ten levels deep, this represents nearly five million potential paths from which to choose during the problem solving process. For real-time problem solving there must be some method for "pruning" this search space. There are two pruning methods predominately used in AES based on the kind of knowledge brought to bear on the problem: Heuristic knowledge and Search knowledge.

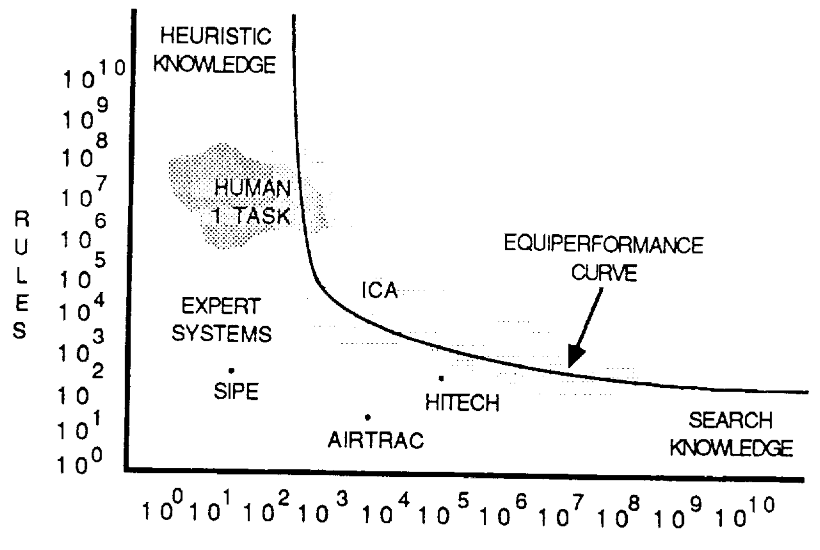
Heuristic knowledge refers to information known to be true due to empirical evidence of operations within the specific domain; an expert system is a good example of capturing this kind of knowledge. Search knowledge refers to information concerning how to traverse a search space to efficiently arrive at a solution to problems; a general problem solving program such as a planning system is a good example of capturing search knowledge. Heuristic knowledge is more useful when the problem solving domain is well defined and search knowledge when it is not. The advantage of heuristic knowledge is that because more information is available at each decision point, selection of the next best step is more precise resulting in fewer overall steps in the problem solving process. The disadvantage is that applying more knowledge at each step could slow down the process and allow for fewer alternatives to be investigated per unit time. Search knowledge is opposite in nature; although it utilizes less information at each decision point making selection of the next best step less precise, more alternatives in the problem solving process can be investigated. The disadvantage is that with less knowledge many more alternatives must be investigated per unit time. Figure 6.1 depicts how AES generally contains a mixture of both kinds of knowledge.

Human experts tend to use more heuristic knowledge while current AES software programs (excluding expert systems) tend to use more search knowledge. For example, it has been estimated that a human with a masters level rating in chess has between 50,000 and 90,000 "chunks" of information, where a chunk is equivalent to approximately 100 rules [13]. Before making a move, a human chess master rarely examines more than 200 board positions, but through heuristic knowledge is able to recognize and evaluate over 50,000 chess patterns. Hitech, on the other hand, examines nearly 20 million board positions before making a move while utilizing less than 200 rules [8]. Using a nearly pure search knowledge approach, Hitech has achieved a master's level chess rating as compared to other expert system chess programs (small search using a large numbers of heuristic rules) which have only achieved a mediocre level of chess play. Hitech exemplifies the first key capability required for ICA systems: a fast and efficient search approach.

Hitech also embodies the second key capability for ICA systems: a compiled knowledge source. Compiled knowledge refers to transforming a declaratively represented knowledge base into an efficient internal representation that then provides quick access to the knowledge. This approach works most effectively with static knowledge and is analogous to compiling a high-level procedural language into machine code. For example, action networks which directly map an agent's goals, beliefs and sensor inputs to a set of effector reactions is an example of compiled knowledge [36]. Another example is Hitech which contains two software programs: an Oracle and a Searcher. The Oracle is a compiled knowledge source which performs a complex analysis of the current board configuration and passes its results to the Searcher program to aid it in performing an efficient search of potential moves. The Searcher utilizes 64 processors operating in parallel, one for each chess board square, to further enable its rapid search of potential moves.

The third key capability for ICA systems is real-time operations on large volumes of error-prone sensor data. Airtrac, the aircraft tracking simulation system, processes and validates 4000 radar tracking reports per second to determine whether an aircraft is coming from or heading for an incorrect destination [34]. This real-time performance is achieved through simulation of 100 processors operating in parallel and utilizing an object-oriented problem solving approach. In this problem solving paradigm, each object represents a specialized problem solver that has its own local knowledge and expertise. Each object is defined in terms of other objects to which it can send messages to solve problems for it. Each object has the ability to store, process and create new information and to communicate with other objects to cooperatively perform problem solving.

The fourth capability is reactive real-time planning. An example of this type of program is SIPE (System for Interactive Planning and Execution monitoring), a simulation of the robot Flakey moving through several rooms [53]. SIPE intermingles planning and execution, and can accept arbitrary unexpected occurrences during execution and modify its plan to take these into account. SIPE's execution speed is demonstrated when Flakey is requested to retrieve an object from one room and deliver it to another. This problem requires the planner to generate hundreds of goal nodes to produce one plan for execution. The complete plan takes about 30 seconds to formulate, but because SIPE intermingles planning and execution, the plan is ready for execution in 9 seconds. The reason for such a long execution time is that SIPE is determining a plan to execute while concurrently monitoring its environment in real-time to determine if that plan should be modified. As shown by SIPE, reactive real-time planning is difficult to implement due to the requirement to effectively interleave planning and plan execution.



SITUATIONS / SECOND

Figure 6.1 Knowledge/Search Trade-Off

Figure 6.1, which was derived from [13], depicts the relative performance of Hitech, Airtrac, SIPE and a human expert and shows the degree of heuristic/search knowledge utilized. The knowledge level is measured in rules and the degree of searching in situations per second. The numeric performance metrics are as follows: Hitech has 200 rules and evaluates on the average 100,000 situations (board positions) per second; Airtrac has 40 rules and evaluates 4000 situations (radar tracking reports) per second; and SIPE has 300 rules and evaluates 17 situations (goal/process nodes) per second. Although each program exhibits a portion of software technology requisite for ICA implementation, they each fall short of expert level performance as defined by the equiperformance curve. This equiperformance curve combines several curves that all represent expert level performance via differing mixtures of heuristic and search knowledge. The gray area surrounding the middle of the equiperformance curve denotes the range of heuristic and search knowledge mixtures that are most probable for ICA programs. Hitech and Airtrac are the closest to ICA performance due to Hitech's efficient search and compiled knowledge and to Airtrac's real-time processing of error prone sensor data. SIPE falls quite short due to implementing real-time reactive planning without incorporating Hitech's and Airtrac's effective performance enhancing techniques.

SIPE's poor performance underscores the need for utilizing fast search, compiled knowledge and real-time processing of large volumes of error-prone sensor data when implementing a real world reactive ICA system. In addition, since ICA applications will be required to perform general purpose planning and may not have sufficient access to heuristic knowledge, a search knowledge approach is more effective. Indeed, attempting to encapsulate heuristic knowledge by enumerating all the potential states of the ICA system is the very thing that must be avoided if ICAs are to exhibit graceful degradation of performance under uncertain and potentially conflicting sensory data.

6.2 Software Technology Requirements

The following subsections present a brief description of the current state-of-the-art in particular software technology fields and then provide a description of advances from this baseline required to support ICA implementation. These specific software technology fields were chosen because they contain functionality vital to application of operational ICA systems. Only software technologies will be discussed; their corresponding hardware technologies, if any, are not.

6.2.1 Natural Language

Natural language encompasses both verbal and written computer understanding and generation. Although these two mediums have different operational requirements, the underlying computer processing to achieve competent performance is basically the same. Current commercial speech recognition systems [41] are limited to recognition of single words or short phrases and require that each person's voice used in the system be "trained" beforehand to allow computer recognition of that voice. In addition, commercial speech recognition systems today only identify and map a recognized word to a programmed action; there is no understanding of what the word or phrase means. Some advances in speech recognition that would facilitate implementing an operational ICA system are:

1) Voice recognition under varying conditions such as inflection and different background noises.
2) Little or no required training for system users.
3) Understanding (as opposed to merely recognition) of verbal commands that may be single words or complete sentences.

In natural language, there are two distinct stages in the language-planning process: deciding *what* to say and deciding *how* to say it [7]. AI natural language systems developed to date for written understanding have focused primarily on the second of these two stages, or language generation. Current techniques for language generation include: grammar-based approaches usually expressed as an augmented transition network [54], utilizing language predicates usually expressed as conceptual dependencies [49], utilizing predetermined "scripts" to glean additional information from words [50], and blackboard approaches which utilize cooperating knowledge bases [18]. Although some progress has been made through these approaches, they all lack in a complete understanding of the *meaning* of what is stated and how it relates to or impacts the environment in which it is used. Additionally, none of these approaches address the problem of deciding *what* should be said.

Some advances in natural language that would facilitate implementing an operational ICA system are:

1) A complete and robust formal theory that permits agents to decide both what to say and how to say it.
2) Integrating written utterances into an understanding of their affects on other agents and the state of the environment.

6.2.2 Model-Based Reasoning

Model-based reasoning uses an internal symbolic model of the system of interest and updates the state of this model based on sensor evidence and cause/effect analysis [44]. This type of reasoning has been used for diagnosis applications and is useful when explicit declaration of all potential system configurations is not a feasible approach to failure resolution. In diagnosis systems, differences between the known internal model of the system and evidence of the state of the actual system form the basis of diagnosis. This form of reasoning may require less knowledge engineering than traditional approaches and is more robust because it can often reason in the case of unanticipated system failures.

At its present state-of-the-art, model based reasoning has several disadvantages. Translating design and system information into an internal model format can be very time consuming and many assumptions often have to be made for the reasoning system to operate correctly. An example of a common assumption in diagnostic model-based systems is of only single failures or of non-intermittent failures. Current model based approaches also require excessive processor time making real-time applications prohibitive.

Some problem areas in model-based reasoning that if solved would facilitate an operational ICA system are:

1) Determining the cause of a difference between the current modeled and sensed value of a sensor, i.e., determining whether or not the sensor has failed, is not a well understood process.
2) High fidelity models that accurately describe complex real-time systems must have efficient methods for updating their internal state in order to maintain rapid changes in the external environment.

3) An increase in model fidelity should not incur a non-linear or exponential increase in model reasoning time.
4) Isolating fault cause(s) when multiple faults occur and when dealing with intermittent failures.

6.2.3 Sensor Fusion

Sensor fusion refers to the process of combining various sensor data streams or their results into an integrated representation. The characteristics of the fusion process is somewhat dependent on the nature of the sensor and its data, examples of which are: color cameras, inertial motion sensors, sonic-imaging sensors, forward-looking infrared imaging, millimeter-wave radar, CO2 laser radar, and others. But in general, three types of sensor fusion can be defined: [24]

1) Competitive -- where sensors provide data that either agrees or conflicts. Both cases arise when sensors provide data of the same form.
2) Complementary -- where sensors provide data of different forms. For example, recognizing three-dimensional objects with two cameras illustrates this kind of fusion.
3) Independent -- where a single sensor is used for separate parts of a task. For example, a laser range finder could be used for short-range landmark recognition while a camera could be used for distant landmark recognition.

Currently, there are at least two approaches to sensor fusion: procedural, ranging from Kalman filtering techniques to feature extraction techniques to full-blown Bayesian observation networks [24]; and the much newer analog neural networks [31], which can perform both data smoothing (noise reduction) and data fusion at much faster rates than serial approaches. Most of the procedural techniques exhibit limited performance due to the complex nature of the data, clutter in the environment and occluded objects. Analog and optical neural network techniques show promise of resolving some of the procedural problems but are still in the investigative stage of development. In addition, there is the seemingly ubiquitous problem of converting fused low-level sensor data into a high-level symbolic interpretation.

Some advances in data fusion techniques that would facilitate implementing an operational ICA system are:

1) Mature optical and analog neural network sensors for vision.
2) Robust and reliable procedural approaches for sensor fusion and conversion of low-level data into high-level representations.

6.2.4 Reasoning with Incomplete Knowledge

Reasoning about action under incomplete information and scarce resources is central to solving difficult problems in AI. Indeed, the very act of preparing knowledge to support reasoning requires that many facts remain unknown, unsaid, or crudely summarized because it is not feasible to enumerate all facts about a domain and especially exceptions to those facts. Currently, there are three major approaches for handling uncertainty: [42]

1) Non-monotonic reasoning -- where standard logical inference is utilized until an inconsistency is detected. When this occurs, facts deduced from previous assumptions are retracted from the knowledge base, new assumptions installed and standard logical inference resumes from the new assumptions.
2) Extensional or calculist -- this approach treats uncertainty as a generalized truth value attached to formulas and (following the tradition of classical logic) computes the uncertainty of any formula as a function of the uncertainties of its subformulas. Examples of this approach are Dempster-Shafer calculus, fuzzy logic and certainty factors.
3) Intensional or probabilist -- this approach treats uncertainty in the framework of probability theory and attaches certainty measures to subsets of "possible worlds" which are combined through set theory operations. Examples of this approach are the Bayesian formalism or an approach which combines Bayesian and Decision Theory [10].

Problems exist with all of these approaches to reasoning under uncertainty. Some researchers feel that non-monotonic reasoning systems are not theoretically sound [55] and it is well known that they are computationally intensive. Extensional uncertainty systems are efficient computationally and easy to use; however, they often yield conclusions that are counterintuitive, and these problems surface in improper handling of bidirectional inference, difficulties in retracting conclusions and improper treatment of correlated sources of evidence [42]. Intensional systems offer the most theoretically sound techniques and can handle problems extensional systems cannot, however, they are computationally expensive because of their global nature. For example, when using the Bayesian formalism, probabilities for all facts conditioned on all other facts are needed a priori; this is difficult, and in some cases, impossible.

Some advances in reasoning with incomplete knowledge that would facilitate implementing an operational ICA system are:

1) Improved computational efficiency of intensional approaches. One approach to improvements may be in belief networks [42].
2) Sound theoretical foundations for extensional approaches.
3) Improved computational efficiency of truth maintenance systems.

6.2.5 Real-Time Reactive Planning

Any intelligent system that operates in a moderately complex or unpredictable environment must be reactive, that is, it must respond dynamically to changes in its environment. A focus of some of the research in this area to date has been in designing software architectures for an autonomous mobile robot [23,27,28,48,53], but the concepts generated are independent of the particular system. Current real-time reactive software architectures are motivated by at least three main desiderata: [26]

1) Modularity -- the system should be built from small components that are easy to implement and understand. This permits testing of simple behaviors first and then allows more complex behaviors to be built on top of them.
2) Awareness -- at no time should the system be unaware of what is happening; it should always be able to react to unexpected sensory data.
3) Robustness -- the system should continue to behave plausibly in novel situations and when some of its sensors are inoperable or impaired.

Current approaches to implementing real-time reactive planning systems for general environments are limited by several problems, most of which are based on the requirement of operations in an unrestricted domain. For example, it has been shown that a general planning problem is undecidable and that even many restricted planning problems are intractable [12]. Another problem, caused more by the requirement of operations in dynamic domains, is that during the planning process the environment may have changed to an extent that the newly created plan is no longer executable in the current situation. Attempts to limit the effects of these problems on real-time operations have so far been ad-hoc domain specific solutions and have not solved the requirement of general operations in dynamic domains.

Some advances in real-time reactive planning approaches that would facilitate implementing an operational ICA system are:

1) Mature parallel software technology for operations such as planning, sensor data assimilation, effector control, etc.
2) Control structures which permit real-time interruption, both scheduled and unexpected, with little impact on current planning and effecting operations. Recent work in this area is the Schemer II architecture currently in progress at the Rockwell Science Center Palo Alto Laboratory [17].

7. Conclusions

This report has described potential AES architectures to support current scenarios which depict operations within the in-space assembly and the manned Mars spacecraft domains. Our approach was to first define the technological barriers to objectives within each domain and then present potential AES solutions to those barriers. The in-space assembly domain was found to have little need for an ICA software system due to the inherently special purpose nature of operations within that domain and due to its heavy dependence on sensor technology which at its current level is deemed too immature to support operational ICA systems. The manned Mars spacecraft domain was again found to have little requirements for ICA software technology, but did have a strong need for AES to support spacecraft autonomy and health maintenance. We conclude then, that within the current-technology time frame for these two domains, ICA software technology is not required.

The major reasons that ICA technology is not recommended, are partly due to the domains covered in the report (obviously, ICA technology is ideal for domains like multiple autonomous rovers) and partly due to the software barriers that must be overcome before it is possible to field operational ICA systems. In the current-technology time frame it was shown that domain operations are very specific and the general purpose reactive behavior of ICA systems is not warranted or cost effective to pursue. Some of the software barriers that must be overcome to support operational ICA systems were also presented; the basic problem is that it is difficult to implement real-time reactive software that can also exhibit general purpose problem solving behavior. Indeed, Nils Nilsson, one of the foremost investigators of ICA software technology today, has stated that there are still many fundamental *conceptual* barriers to overcome before even prototype ICA systems are realizable [38].

Although ICA software technology cannot be recommended, this report has recommended and provided preliminary AES architectures that could fulfill currently defined operational requirements for these two domains. Within in-space assembly, astronaut extra-vehicular activities in conjunction with televideo and/or telepresence robotic technology was deemed sufficient to assemble exploration-class spacecraft. In the manned Mars spacecraft domain, a detailed AES architecture was presented that could automate spacecraft operations, perform subsystem health maintenance and fault detection, isolation and reconfiguration, and provide some level of decision support for its crew when communication with mission control is not possible or is time prohibitive. It is felt that this software support of spacecraft operations is crucial for providing crew ease in daily activities and increases the probability of mission success.

Another conclusion that can be drawn from this research is that as exploration-class missions become more advanced and autonomy requirements increase, ICA software technology could become an important factor in determining the types of missions that may be attempted. Indeed, as manned and unmanned space exploration becomes more remote and autonomous, there will be a greater need for the reactive general problem solving capability that would be exhibited by operational ICA systems. For ICA technology to be realizable when other hardware and software technology matures, research into solving the barriers limiting operational ICA systems should begin now. A further benefit of this research would be the necessary integration of disparate AI disciplines, resulting in a synergism and advancing AI software technology to a level that should enable it to solve some of the challenges of space exploration missions in the 21st century.

To achieve the goal of operational ICA systems to support space exploration missions in the projected-technology time frame, research should be directed in several study areas. First, potential AES architectures for space operations should be defined as scenarios are developed. A starting point for this effort is provided in Appendix 8.2. Early definition of the software requirements and architectures to support these space scenarios is necessary to insure that an integrated systems engineering approach is utilized to support their development. Second, the ICA research at Stanford University should be continued so that the target of a prototype ICA system testable in a simulated environment is achieved. Early investigation into producing viable ICA architectures will insure that a mature software technology is available when it is needed. Lastly, Section 5 of this report has defined a preliminary AES architecture for the manned Mars spacecraft. A detailed study of the software issues associated with this concept, including architecture prototyping to refine requirements and cost estimates, should be initiated. It is important to begin this research now, so that software requirements and design form an integral part of the systems engineering of the Mars spacecraft and its support systems.

8. Appendices

8.1 The Apollo 13 Accident

The following is a condensed transcription of testimonies given by Dr. Rocco A. Petrone, Apollo Program Director, Glynn S. Lunney, Apollo 13 Mission Director, James A. Lovell, Commander of Apollo 13 and John L. Swiggert, Command Module Pilot of Apollo 13 before the Committee on Aeronautical and Space Sciences United States Senate, Ninety-first Congress Second Session on April 24, 1970.

1) Apollo 13 mission anomaly assessment

When Commander Lovell of the Apollo 13 mission to the moon radioed to Mission Control Center the detection of an anomaly aboard the spacecraft (an O2 tank on-board the service module had exploded), the awareness of the problem went through a series of stages that ranged from anticipation that the lunar landing mission, that is landing on the moon, might be in jeopardy to one of whether or not the astronauts would be able to return back to Earth.

2) Experts gathered

In accordance with general practice, a large number of contractor systems personnel were providing direct support to the mission at all times. In order to bring all available resources to bear, the major spacecraft contractors, their subcontractors and vendors were immediately requested by ground control to augment this around-the-clock support.

3) Situation recreated

Recreation of the incident by using the telemetry radioed back to Earth had not successfully modeled the actual cause of the problem by the time of the Congressional hearing. However, this deep analysis was not necessary for the successful recovery of the ship and crew.

4) Sufficient analysis for the situation performed

With regard to the incident itself and its cause, the immediate analytical efforts during the mission were concentrated only on the analysis necessary to determine effects or delayed effects on other subsystems. The prime objective was to ensure that all necessary action was being taken to bring the astronauts back to Earth as quickly as possible with the least possible imposed additional risk.

5) Recommendations made

In about an hour and a half both the ground controllers and the astronauts realized that the astronauts' only hope of survival was for them to go into the lunar module, Aquarius, power up its systems, and use the systems and the consumables on-board for the return home. It was here that the astronauts realized that it was now a case of survival. The lunar module was to be used as a lifeboat.

When the astronauts had powered up the lunar module, the next milestone was to use the lunar module's controls to maneuver the spacecraft. The lunar module is not normally used to control the entire stack of crafts; this is usually done from the command module. They were, at that time, in a trajectory that would have entered them into a permanent orbit around the moon of 230 miles apogee and 100 miles perigee. The ground controllers gave the astronauts the information they needed to make a free return trajectory burn to get them back to Earth.

The flexibility of the spacecraft systems provided the ground controllers with many different options from which they selected a series of configurations in meeting the varying requirements on the trip home. This long, arduous voyage continually presented the challenge to balance the spacecraft systems required to perform necessary functions against the availability of consumables of water, electrical power, oxygen, and the lithium hydroxide used to remove carbon dioxide.

One of the critical consumables was the water, and the astronauts were not going to use any of the lunar module water because it was too scarce. With the help of the ground controllers they devised a procedure for extracting the water from the potable tank in the command module. Approximately 24 hours prior to reentry they ran out of water in the potable tanks in the command module.

One of the problems the astronauts had to solve was removing excess carbon dioxide from the lunar module air. Ground controllers read up the instructions and astronauts Lovell and Swiggert constructed a lithium hydroxide canister which worked perfectly. It reduced the partial pressure of the carbon dioxide from 8.5 mm down to 0.2 millimeter in 2.5 hours.

After the second maneuver, which was a burn within about 130 miles of the lunar surface designed to decrease the astronauts' time returning home, a third maneuver was required. The astronauts found out from ground control that their angle for reentering the Earth's atmosphere was insufficient. Mission control continued to give them the procedures for this third maneuver. The third maneuver was different because they were powered down, essentially a dead spacecraft except for communications and the life support system, particularly the air and oxygen pumps keeping things cool. This maneuver was to line up the spacecraft's attitude with the Earth's terminator, which was a handy reference system. All three men had to help in executing this manual burn under the direction of mission control.

There were batteries on board the command module necessary for reentry that were weak. Mission control had previously provided the astronauts with a procedure to recharge the batteries from the lunar module power system. But because of their fatigued state, astronaut Lovell requested that mission control come up with another procedure that the astronauts could do accurately in their fatigued condition without any mistakes. Ground control produced the procedures to power up the command module, jettison the lunar module, perform a final alignment of the command module and make a normal reentry.

6) Parallels between the Apollo 13 and a Mars expedition

Consider how the situation the Apollo 13 astronauts encountered during their ill-fated trip to the moon would have developed if the same mishap had occurred just as the Mars orbiter passed behind Mars as seen from Earth. In this worst-case scenario, we will assume the planets are nearly in major opposition (the sun is directly between them). At this distance light will take 42 minutes to pass from Mars to Earth and back to Mars. Also, the spacecraft will be occulted for approximately one hour when its orbit takes it behind Mars. So, even if Ground Control Center did not need any time to come up with a solution, the total time from the transmission of the recognition of the problem to receipt of instructions from Earth would be about 97 minutes.

The following is a selection of the sequence of events that occurred during the Apollo 13 mission. Interspersed (in capital letters) through the log are indications of how this situation would affect a manned Mars orbiter. Leeway must be granted in the significance of the details of this log as they would apply to the Mars orbiter. What should be noted is how time would work against a viable solution to such a problem.

ELAPSED TIME (from recognition of problem)	EVENT
-00:02:20	Oxygen tank fans turn on.
-00:01:58	High current spike, fuel cell #3.
-00:01:44	Oxygen tank #2 pressure rise.
-00:01:42	11.3 volt transient, AC bus #2.
-00:01:39	High current spike, fuel cell #3.
-00:01:35	Oxygen tank #2 temperature rise.
-00:00:35	Oxygen tank #2 maximum recorded pressure.
-00:00:27	Measurable spacecraft motion.

-00:00:24	Oxygen tank #2 pressure zero.
00:00:00	Lovell "I believe we have a problem here." AT THIS POINT THE MARS ORBITER DISAPPEARS BEHIND MARS.
00:57:00	THE MARS ORBITER REAPPEARS FROM BEHIND MARS. THE CREW RADIOS EARTH FOR HELP. THEY HAVE 57 MINUTES TO DESCRIBE THE SITUATION AS BEST THEY CAN. SINCE THEY ARE NOT IN DIRECT CONTACT WITH SYSTEMS EXPERTS, THEY WILL HAVE TO ASSESS THE SITUATION THEMSELVES.
01:18:00	EARTH RECEIVES MESSAGE FROM MARS.
01:40:00	AN INSTANT REPLY FROM EARTH REACHES MARS. THE MESSAGE CAN BE RECEIVED BY THE MARS ORBITER FOR 20 MINUTES BEFORE THE SPACECRAFT DISAPPEARS BEHIND MARS AGAIN. IF THE GROUND CREW TAKES 15 MINUTES TO DELIBERATE, THE MESSAGE CAN BE 5 MINUTES LONG. IF MORE THAN 20 MINUTES ARE NEEDED TO RESPOND, THE MARS ORBITER CREW WILL HAVE TO WAIT TWO HOURS FROM START OF THEIR TRANSMISSION TO RECEIVE THE ANSWER FROM EARTH - THAT IS THREE HOURS FROM THE FIRST DETECTION OF TROUBLE.
01:41:40	Upon recommendation from ground control, the crew enters into the lunar module.
02:00:00	THE SPACECRAFT DISAPPEARS BEHIND MARS.
02:14:40	Transfer guidance alignment to lunar module. All navigation will be done from the lunar module, which is an unusual configuration.
02:44:40	Command module power down.
03:00:00	THE SPACECRAFT REAPPEARS FROM BEHIND MARS. IF EARTH HAD RECEIVED ENOUGH INFORMATION FROM THE ORBITER'S FIRST TRANSMISSION TO SOLVE THE PROBLEM, EARTH'S TRANSMISSION CONTAINING THE SOLUTION CAN BE RECEIVED BY THE ORBITER. OTHERWISE ANOTHER MESSAGE MUST BE SENT FROM MARS.
05:34:40	Decision - execution of midcourse for free return. THE ASTRONAUTS WOULD HAVE TO WAIT ABOUT 25 MINUTES TO GET THIS INFORMATION.
07:24:40	A decision on projected overall flight plan. IF TIME WERE OF THE ESSENCE, THE ASTRONAUTS WOULD EFFECTIVELY HAVE TO BE LEFT OUT OF THE DECISION MAKING PROCESS, SINCE TURN-AROUND TIME IS SO LONG.
23:32:40	Decision - execution of maneuver 2 hours after closest approach to moon.
38:04:40	Carbon dioxide successfully removed with lithium hydroxide canister fix. THE INSTRUCTIONS ON THE METHODS FOR THE FIX WOULD HAVE TO BE GIVEN, AT BEST, IN 57 MINUTE CHUNKS. THE INSTRUCTIONS WOULD HAVE TO BE EXPLICIT ENOUGH SO THAT NO QUESTIONS WOULD BE RAISED AS THE PROCEDURE PROGRESSED. FEEDBACK FROM THE ASTRONAUTS WOULD BE ZERO. IF A SITUATION EXISTED THAT HINDERED THE PROCESS, THAT WOULD HAVE

TO BE RADIOED TO EARTH AND THE FIX WOULD HAVE TO WAIT FOR THE NEXT CYCLE.

A spacecraft as complex as the manned Mars orbiter must be expected to incur subsystem failures over the duration of its mission. Since many potential failures can prove life-threatening if left unresolved for even a few seconds, the three hour delay discussed above is clearly unacceptable.

Astronauts can be trained to handle the most likely of the critical failures, but even the best training will fade if not reinforced regularly through practice and drills. In a complex vessel with a small crew, the number and complexity of drills required to maintain expertise in contingency procedures, could become a major obstacle to other mission objectives. Also, the astronauts could run the risk of becoming lax in the routine execution of contingency procedures which could lead to disastrous results.

8.2 Suggestions For Further Study

Since research was begun on this report in late 1988, many new advanced manned and unmanned space operational scenarios have been defined by NASA Ames. Each of these scenarios warrant further study but are beyond the scope and intended purpose of this report. Therefore, we provide only a brief description of the types of software architectures required to implement each of these scenarios.

I. Lunar Evolution Configuration: Low Earth Orbit Transportation Node -- Assembly of the Lunar Vehicle Hangars. These two software modules are generic for any space assembly process.

1. An expert system to control and monitor the assembly process by providing:

   a. Step-by-step procedures for telepresence operators.
   b. Contingency operations in the event of a mishap or for work-arounds.

2. Real-time procedural software (tasked Ada) to perform:

   a. Telepresence and robotics operations.
   b. Processing of sensor data for structural integrity checkout.
   c. Rapid attitude adjustments to the transportation node to compensate for vibration and inertial reactions to large mass movements during the assembly process.

II. Mars Evolution Configuration: Low Earth Orbit Transportation Node -- Assembly of the Manned Mars Vehicle:

1. Requires the two generic assembly software modules described in I above.

2. Once the vehicle is assembled, vehicle checkout and test requires an expert system to interpret sensor and BITE data for fault detection, isolation and vehicle reconfiguration.

III. Propellant Transfer in 0-g Orbit:

1. Requires a feedback control loop system to adjust the internal screw velocity based on sensor data of void development inside the propellant cylinder.

2. Trend data analysis of void developments could be performed by procedural software to increase smooth propellant transfer.

IV. Automated Docking via Robotic Arms:

1. Requires a real-time procedural software system to:

   a. Process CCD TV data for identification of and guidance toward docking target.
   b. Identify hazardous conditions based on visual data.
   c. Provide GNC for docking operations (using LIDAR and near field electromagnetic guidance).
   d. Update level of criticality for docking process based on estimated time till docking.
   e. Broadcast appropriate warnings based on this criticality level.

V. Tended Lunar Node (Propellant Depot) Docked to the Lunar Ascent/Descent Vehicle:

1. Requires the two generic docking software modules described in IV above.

2. Requires a hybrid expert system (heuristics and model-based reasoning) to perform health maintenance and automated checkout to:

   a. Report problems discovered during inactive periods to Space Station or Lunar personnel.
   b. Perform checkout before proceeding with a major operation.

VI. Helical Fuel Tank Loader for Low Orbit Refueling:

1. Requires the two propellant transfer modules described in III above.

2. Requires the two generic docking software modules described in IV above

VII. Mars/Lunar Automated Assembly of Geodesic Dome:

1. Requires multiple mobile robots with ICA software to perform:

   a. Manual and tandem operations such as fastening, coordinating, planning, commanding, etc.
   b. Replanning due to unforeseen events such as replacing defective parts or assisting a disabled robot.
   c. On-site incremental inspection of structural integrity.

VIII. Space Tug: Fuel Tank Transport and Structure Assembly:

1. Requires procedural software to perform:

   a. GNC, subsystem management, avionics, sensor control.
   b. Obstacle avoidance.

2. A hybrid expert system to perform:

   a. Subsystem health test and monitor sensors for trend data analysis.
   b. Fault detection, isolation and reconfiguration of subsystems.

The above outline describing software concepts for these eight space operations scenarios is preliminary and high-level. Additional research is required to fully define software architectures for these systems.

8.3 References

1) D. L. Akin, M. L. Minsky, E. D. Thiel and C. R. Kurtzman, *Space Applications of Automation, Robotics and Machine Intelligence Systems (ARAMIS) - Phase II, Volume 1: Telepresence Technology Base Development*, Massachusetts Institute of Technology, NASA Contractor Report 3734, 1983.

2) D. L. Akin, M. L. Minsky, E. D. Thiel and C. R. Kurtzman, *Space Applications of ARAMIS - Phase II, Volume 2: Telepresence Project Applications*, MIT, 1983.

3) D. L. Akin, M. L. Minsky, E. D. Thiel and C. R. Kurtzman, *Space Applications of ARAMIS - Phase II, Volume 3: Executive Summary*, MIT, 1983.

4) Ames Research Center, *Special Assessment Studies: Automation and Robotics/Human Performance for Exploration Class Missions*, NASA Ames Research Center, July, 1988.

5) Ames Research Center, *Systems Autonomy Technology Program Plan - Executive Summary*, NASA Ames Research Center, December 1987.

6) Ames Research Center, *Systems Autonomy Technology Program Plan - Report*, NASA Ames Research Center, December 1987.

7) Douglas E. Appelt, *Planning Natural-Language Utterances to Satisfy Multiple Goals*, Technical Note 259, Artificial Intelligence Center, SRI International, March 1982.

8) H. J. Berliner and C. Ebeling, *The SUPREM Architecture*, Artificial Intelligence 28(1), 1986.

9) Boeing Aerospace Corporation, *Engineering Analysis for Assembly and Checkout of Space Transportation Vehicles in Orbit*, Final Briefing Presentation, October 1988.

10) John S. Breese, Eric J. Horvitz and Max Henrion, *Decision Theory in Expert Systems and Artificial Intelligence*, International Journal of Approximate Reasoning, 2:247-302, 1988.

11) M. J. Buckley, M. R. Fehling, D. E. Smith and M. L. Ginsberg, *Resource Bounded Intelligent Systems*, Fiscal Year 1989 IR&D Technical Plan, Rockwell International Science Center, 1989.

12) David Chapman, *Planning for Conjunctive Goals*, Technical Report 802, Artificial Intelligence Laboratory, MIT, 1985.

13) W. G. Chase and H. A. Simon, *The Mind's Eye in Chess*, Visual Information Processing, Academic Press, 1973.

14) Philip R. Cohen, *Persistence, Intention, and Commitment*, Technical Note 415, Artificial Intelligence Center, SRI International, February 1987.

15) Philip R. Cohen, *Rational Interaction as the Basis for Communication*, Technical Note 433, Center for the Study of Language and Information, SRI International, April 1988.

16) Committee on Aeronautical and Space Sciences, *Hearing before the Committee on Aeronautical and Space Sciences*, United States Senate, 91st Congress, April 24, 1970.

17) Michael R. Fehling and John S. Breese, *A Computational Model for Decision-Theoretic Control of Problem Solving Under Uncertainty*, Technical Memorandum 837-88-5, Rockwell Science Center Palo Alto Laboratory, April 1989.

18) Richard D. Fennell and Victor R. Lesser, *Parallelism in Artificial Intelligence Problem Solving: A Case Study of Hearsay II*, Readings in Distributed Artificial Intelligence, Edited by Alan H. Bond and Les Gasser,

Morgan Kaufmann Publishers, Inc., 1988.

19) Kenneth W. Fertig, *Distributed Intelligent Systems*, IR&D Technical Plan 840, Science Center Laboratory, Rockwell International, September 1988.

20) Michael R. Genesereth, *The Use Of Design Descriptions in Automated Diagnosis*, Artificial Intelligence 24 (1984) 411-436.

21) Michael P. Georgeff, *Many Agents Are Better Than One*, Technical Note 417, Representation and Reasoning Program, SRI International, March 1987.

22) Michael P. Georgeff, *Planning*, Technical Note 418, Representation and Reasoning Program, SRI International, March 1987.

23) Michael P. Georgeff, Amy L. Lansky and M. J. Schoppers, *Reasoning and Planning in Dynamic Domains: An Experiment with a Mobile Robot*, Technical Note 380, Artificial Intelligence Center, SRI International, 1987.

24) Yoshimasa Goto and Anthony Stentz, *Mobile Robot Navigation: The CMU System*, IEEE Expert, Winter 1987.

25) Ewald Heer, *Private Communication*, June 1988.

26) Leslie Kaebling, *An Architecture for Intelligent Reactive Systems*, Technical Note 400, Computer and Information Sciences Division, SRI International, October 1986.

27) Theodore A. Linden, James P. Marsh and Doreen L. Dove, *Architecture and Early Experience with Planning for the ALV*, IEEE International Conference on Robotics and Automation, San Francisco CA, April 7-10 1986.

28) Theodore A. Linden and Jay Glicksman, *Contingency Planning for an Autonomous Land Vehicle (ALV)*, Proceedings of the IJCAI, 1987.

29) Theodore A. Linden and Sam Owre, *Transformational Synthesis Applied to ALV Mission Planning*, DARPA Knowledge Based Planning Workshop, Austin TX, December 8-10, 1988.

30) Martin Marietta Corporation, *Case Study 2: Human Expedition to Mars*, Final Briefing Presentation, July 1988.

31) B. P. Mathur, *Analog Sensor Fusion for Target Identification*, Fiscal Year 1989 Technical Plan, Rockwell International Science Center, 1989.

32) Louis S. McTamaney, *Mobile Robots: Real-Time Intelligent Control*, IEEE Expert, Winter 1987.

33) G. Montgomery, J. Elder and P. Hess, *Abductive Reasoning Applied to Uncertainty in Diagnostics*, Artificial Intelligence in Armament Workshop, March 21-24, 1988.

34) Russell Nakano and Masafumi Minami, *Experiments with a Knowledge-Based System on a Multiprocessor*, Report No. 87-61, Knowledge Systems Laboratory, Stanford University, October 1987.

35) The Report of the National Commission on Space, *Pioneering the Space Frontier*, Bantam Books, May 1986.

36) Nils Nilsson, *Action Networks*, Working Paper, Computer Science Department, Stanford University, August 1988.

37) Nils Nilsson, Philip Cohen, Stanley Rosenschein and Kenneth Fertig, *Intelligent Communicating Agents*, Proceedings of the Second Annual AI Research Forum, November 1987.

38) Nils Nilsson, *Private Communication*, August, 1988.

39) Nils Nilsson, *Research on Distributed Cooperating AI Systems*, Computer Science Department, Stanford University, October 1986.

40) G. A. Nixon, D. DeLaquil, T. Reid, B. Rejai and D. Zeilingold, *Vehicle On-Board Expert Systems*, Advanced Launch System Proposal - Technology Demonstration Section, Rockwell International, Expert Systems Application Group, April 1988.

41) J. E. Ogden, W. J. Sabatine, F. J. Skomial and J. D. Thompson, *Man-Machine Interface Study*, Fiscal Year 1985 IR&D Final Report, Rockwell International, Expert Systems Application Group, September 1985.

42) Judea Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, Inc., 1988.

43) Raj Reddy, *Foundations and Grand Challenges of Artificial Intelligence*, AI Magazine, Winter 1988.

44) Raymond Reiter, *A Theory of Diagnosis from First Principles*, Readings in Nonmonotonic Reasoning, Edited by Matthew L. Ginsberg, Morgan Kaufmann Publishers, Inc., 1987.

45) B. Rejai, D. Zeilingold and V. Eng, *Long Duration Orbiter Safing and Failure Detection Expert System Study*, Fiscal Year 1988 IR&D Final Report, Rockwell International, Expert Systems Application Group, September 1988.

46) Jeffrey S. Rosenschein and Michael R. Genesereth, Communication and Cooperation, Report No. HPP-84-5, Heuristic Programming Project, Stanford University, March 1984.

47) Jeffrey S. Rosenschein, *Rational Interaction: Cooperation Among Intelligent Agents*, Ph.D. Thesis, Department of Computer Science, Stanford University, October 1985.

48) M. J. Schoppers, *Universal Plans for Reactive Robots in Unpredictable Environments*, Proceedings of the IJCAI, 1987.

49) Roger C. Schank and Christopher K. Reisbeck, *Inside Computer Understanding*, Lawrence Erlbaum Associates, Publishers, 1981.

50) Roger Schank and Robert Abelson, *Scripts, Plans, Goals and Understanding*, Lawrence Erlbaum Associates, Publishers, 1977.

51) *Toward a Unified Theory of Cognition*, Science 241:28, July 1988.

52) David E. Wilkins, *High-Level Planning in a Mobile Robot Domain*, Technical Note 388, Artificial Intelligence Center, SRI International, July 1986.

53) David E. Wilkins, *Practical Planning: Extending the Classical AI Planning Paradigm*, Morgan Kaufmann Publishers, 1988.

54) Patrick Henry Winston, Berthold Klaus and Paul Horn, *Lisp, Second Edition*, Addison-Wesley Publishing Co., 1984.

55) David J. Israel, *What's Wrong With Non-Monotonic Logic?*, Readings in Nonmonotonic Reasoning, Edited by Matthew L. Ginsberg, Morgan Kaufmann Publishers, Inc., 1987.